CCSLM as a Linear Algebra Playground

A technical note for linear algebra researchers

1. The basic object

Concatenated Context-Selected Linear Mappings (CCSLM) can be viewed without reference to neural networks as a problem in matrix composition.

The basic system is

$$\boxed{y = L_n(C_n, x_n) \cdots L_2(C_2, x_2) L_1(C_1, x_1) x}$$

where:

- x is the input vector.
- L_i is a linear mapping at stage i.
- C_i is contextual information available at that stage.
- The context determines, selects, or modulates the linear mapping.
- The resulting mappings are concatenated through ordinary matrix multiplication.

The important conceptual shift is simple:

> **Instead of studying one fixed matrix product, study a family of matrix products indexed by context.**

This immediately turns CCSLM into a broad collection of familiar linear-algebra problems.

---

2. The smallest interesting experiment

Consider three stages, each with two possible matrices:

$$L_1 \in \{A_1, B_1\}, \qquad L_2 \in \{A_2, B_2\}, \qquad L_3 \in \{A_3, B_3\}.$$

There are only eight possible computational states:

$$A_3A_2A_1, \quad A_3A_2B_1, \quad A_3B_2A_1, \quad A_3B_2B_1,$$
$$B_3A_2A_1, \quad B_3A_2B_1, \quad B_3B_2A_1, \quad B_3B_2B_1.$$

A useful first experiment is simply to enumerate these products and examine:

- rank
- determinant
- trace
- eigenvalues
- singular values
- spectral radius

- Frobenius norm
- operator norm
- condition number.

The interesting question is not merely that there are eight combinations.

It is:

> **How many computationally distinct transformations do those eight combinations actually produce?**

---

3. Combinatorial explosion versus matrix-space redundancy

With n stages and k choices per stage,

$$N = k^n.$$

The apparent number of possible computational paths therefore grows exponentially.

But the resulting matrices need not be exponentially diverse.

Different context sequences can produce:

$$M_a = M_b$$

or

$$\|M_a - M_b\| \ll 1.$$

They can also be related by:

- shared singular subspaces
- similar eigenvalue spectra
- low-rank perturbations
- approximate symmetries
- commuting or nearly commuting factors.

This suggests a fundamental CCSLM research problem:

$$\boxed{\text{context sequences} \longrightarrow \text{distinct points in matrix space}}$$

rather than simply counting context combinations.

---

4. A matrix atlas

For every context state C, define its resulting transformation:

$$C \longrightarrow M_C.$$

For example,

$$M_C = L_n(C_n) \cdots L_1(C_1).$$

The collection

$$\mathcal{M} = \{M_C : C \in \mathcal{C}\}$$

can be regarded as a **computational matrix atlas**.

A simple distance between computational states is

$$d(M_a, M_b) = \|M_a - M_b\|_F.$$

One can therefore construct a distance matrix

$$D_{ij} = \|M_i - M_j\|_F$$

and apply familiar tools:

- hierarchical clustering
- multidimensional scaling
- PCA
- spectral methods
- nearest-neighbour analysis.

This gives a direct way to investigate whether the context space induces meaningful structure in computational space.

---

5. Context Specialization

One useful experimental quantity is the degree to which context actually changes the computation.

For example, define

$$d_{ij} = \frac{\|M_i - M_j\|_F}{\max(\|M_i\|_F, \|M_j\|_F) + \epsilon}.$$

A collection of contexts for which d_{ij} is consistently small is computationally **context-insensitive**.

Conversely, large distances indicate strong context specialization.

This suggests a useful empirical question:

> **Does a trained or designed CCSLM develop a small number of computational clusters, or does it occupy a large and diverse region of matrix space?**

The same idea can be extended to singular-value spectra, eigenspaces, or other matrix invariants.

---

6. Noncommutativity

CCSLM also provides a very simple laboratory for studying the consequences of noncommuting transformations.

Compare

$$ABC$$

with

$$ACB.$$

Their difference is related to the commutator

$$[A, B] = AB - BA.$$

When the selected matrices commute, changing their order may have little or no effect.

When they do not commute, **context can effectively select different trajectories through matrix space merely by changing the order of operations.**

This suggests experiments involving:

- commutators
- nested commutators
- matrix groups
- Lie algebras
- products of rotations
- products of shears.

CCSLM therefore provides a computationally accessible route from ordinary matrix multiplication into richer operator-algebraic questions.

---

7. Orthogonal CCSLM

A particularly clean experimental class is obtained by restricting every mapping to the orthogonal group:

$$L_i^T L_i = I.$$

Then

$$M_C^T M_C = I$$

for every complete context.

Consequently, there is no change in Euclidean norm:

$$\|M_C \boldsymbol{x}\| = \|\boldsymbol{x}\|.$$

The problem becomes one of studying **context-selected rotations and reflections**.

Useful experiments include:

- eigenvalue distributions
- rotation angles
- invariant subspaces
- distances between orthogonal matrices
- products of reflections
- products involving Hadamard matrices
- trajectories on $O(n)$ or $SO(n)$.

This is particularly attractive because numerical conditioning is no longer the primary complication.

---

8. Shear CCSLM

Elementary shears provide another excellent test case.

For example,

$$S_x(a) = \begin{pmatrix} 1 & a & 0 \\ & 1 & \end{pmatrix}, \qquad S_y(b) = \begin{pmatrix} 1 & 0 & \\ b & 1 & \end{pmatrix}.$$

Context can select the parameters:

$$C \to S_x(a_C)$$

or select the entire operator.

Products of such simple matrices can generate surprisingly complicated transformations.

A useful visualization is to apply the resulting matrix to:

- a square
- a Cartesian grid
- a circle
- a point cloud
- an image.

This provides an unusually tangible demonstration of the central CCSLM idea:

> **Simple linear operators can produce a rich computational family when their selection and composition are context-dependent.**

---

9. Singular-value experiments

For each context,

$$M_C = U_C \Sigma_C V_C^T.$$

The singular values provide an immediate description of how that context transforms space.

One can examine:

- expansion
- contraction
- effective rank
- anisotropy
- numerical conditioning.

For example,

$$\kappa(M_C) = \frac{\sigma_{\max}(M_C)}{\sigma_{\min}(M_C)}$$

provides a context-dependent condition number.

This raises an interesting systems question:

> **Can context selection systematically move a computation toward better-conditioned transformations?**

That question is entirely accessible using standard numerical linear algebra.

---

10. Low-rank contextual modulation

CCSLM need not require completely independent matrices.

A mapping could be represented as

$$L_C = L_0 + \Delta L_C$$

with

$$\Delta L_C = U_C V_C^T$$

of low rank.

This gives a potentially useful experimental bridge between CCSLM and low-rank parameterization.

Questions include:

- How much contextual diversity can a rank-r modification produce?
- How does r affect the dimension of computational space?
- Can many contexts share L_0?
- When does low-rank modulation become indistinguishable from selecting a completely different matrix?

---

11. Perturbation and sensitivity

Let

$$M = L_n \cdots L_k \cdots L_1.$$

Perturb one stage:

$$L_k \to L_k + \Delta L_k.$$

To first order,

$$\Delta M \approx L_n \cdots L_{k+1}\, \Delta L_k\, L_{k-1} \cdots L_1.$$

This makes CCSLM an interesting setting for studying **where computational sensitivity resides in a matrix cascade**.

One can experimentally determine:

- which stages dominate perturbation
- how sensitivity depends on context
- how conditioning propagates
- whether certain contexts suppress or amplify perturbations.

---

12. Structured matrices

CCSLM experiments need not use dense random matrices.

Interesting matrix families include:

- diagonal matrices
- permutation matrices
- sparse matrices
- circulant matrices
- Toeplitz matrices
- orthogonal matrices
- Householder reflections
- Givens rotations
- Hadamard matrices
- Fourier transforms
- shear matrices
- low-rank matrices.

Each family gives a different computational geometry.

For example, a CCSLM constructed entirely from Givens rotations becomes a context-dependent sequence of planar rotations.

One constructed from sparse matrices becomes a context-dependent sparse computation.

One constructed from permutation matrices becomes a context-dependent reordering system.

---

13. A particularly interesting question: expressive diversity

Given a library

$$\mathcal{L} = \{L_1, L_2, \dots, L_k\},$$

and a depth n, how large is the effective set

$$\mathcal{M}_n = \{L_{i_n} \cdots L_{i_2} L_{i_1}\}?$$

The raw upper bound is

$$k^n.$$

But numerical equivalence, symmetries, and algebraic relations may reduce this dramatically.

Measure:

$$\operatorname{rank}(\mathcal{M})$$

in an appropriate vectorized representation, or cluster the matrices using

$$\operatorname{vec}(M).$$

This gives an empirical notion of **computational diversity versus depth**.

---

14. Software mini-projects

CCSLM is particularly easy to explore using existing linear-algebra environments.

MATLAB

A small experiment can:

- generate matrix libraries
- enumerate context sequences
- multiply matrices
- compute `svd`, `eig`, `rank`, `cond`
- cluster the resulting matrices
- visualize transformed grids.

Python / NumPy / SciPy

Useful for:

- large random experiments
- batched matrix multiplication

- clustering
- dimensionality reduction
- interactive notebooks.

Julia

Particularly attractive for:

- high-performance matrix experimentation
- multiple dispatch over matrix types
- sparse and structured matrices
- large parameter sweeps.

Mathematica

Excellent for:

- symbolic matrix products
- exact algebra
- commutators
- eigenvalue analysis
- simplifying products
- experimenting with parameterized contexts.

SageMath

Useful when the experiment moves toward:

- exact algebra
- matrix groups
- finite fields
- symbolic computation
- group-theoretic structure.

GPU frameworks

A natural computational experiment is to generate enormous batches of

$$M_C = L_n(C_n)\cdots L_1(C_1)$$

and study the resulting matrix distribution.

This turns CCSLM into a **batched matrix-product benchmark** as well as a theoretical experiment.

---

15. A minimal CCSLM laboratory

A useful general-purpose experiment needs only five components:

1. **Matrix library**

$$\{A_{ij}\}$$

2. **Context generator**

$$C \to (i_1, i_2, \ldots, i_n)$$

3. **Matrix composer**

$$M_C = L_n \cdots L_1$$

4. **Matrix analyser**

5. rank

6. norm
7. SVD
8. eigenvalues
9. condition number

10. determinant

11. **Matrix-space visualizer**

12. pairwise distance

13. clustering
14. singular spectra
15. eigenvalue plots
16. transformed geometry.

This is enough to turn CCSLM into an experimental playground without implementing a neural network at all.

---

16. Why this may interest linear algebra researchers

CCSLM reframes several familiar objects as components of a larger problem:

- **matrix products** become context-dependent computational states;
- **matrix groups** become possible libraries of computational operators;
- **commutators** measure order sensitivity;
- **SVD** describes context-dependent geometric distortion;
- **conditioning** describes numerical quality of computational states;
- **low-rank perturbations** provide efficient contextual modulation;
- **clustering** reveals computational equivalence classes;
- **random matrices** provide statistical baselines;
- **structured matrices** provide interpretable computational primitives.

The interesting object is therefore not necessarily a new matrix operation.

It is the **family of matrix products generated by context**.

---

17. A suggested first experiment

For anyone wanting to spend only an afternoon on CCSLM:

1. Choose 4×4 matrices.
2. Create four matrix pairs:

$$(A_1, B_1), \ldots, (A_4, B_4).$$

3. Enumerate all $2^4 = 16$ contexts.
4. Calculate the 16 resulting matrices.
5. Calculate pairwise Frobenius distances.
6. Cluster them.
7. Plot their singular values.
8. Repeat using:
9. random matrices
10. orthogonal matrices
11. shears
12. low-rank perturbations.
13. Compare the resulting computational diversity.

The experiment requires almost no machinery beyond ordinary linear algebra.

Yet it immediately raises deeper questions about:

> **composition, context, matrix-space geometry, computational diversity, redundancy, conditioning, and expressivity.**

That is perhaps the most useful way to approach CCSLM from a linear algebra perspective:

$$\boxed{\text{Context} \rightarrow \text{matrix selection} \rightarrow \text{matrix composition} \rightarrow \text{a point in computational matrix space}}$$

The neural-network interpretation can come later. The linear algebra is already interesting on its own.